project2_109550206

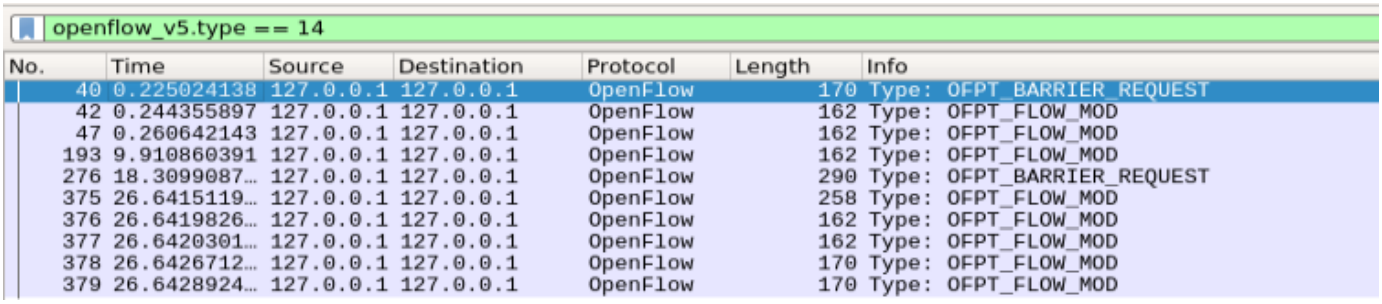
軟體定義網路及網路功能虛擬化 Lab2
ID: 109550206 Name:陳品劭
Self link (https://hackmd.io/@pinchen/SDNFVproject2)

Part 1

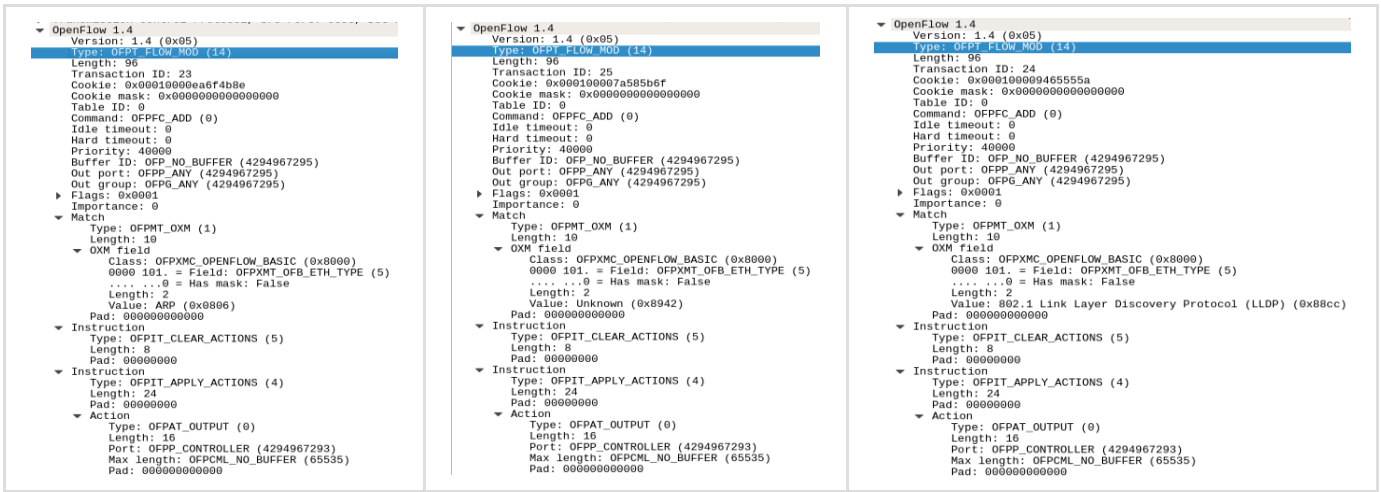
1. How many OpenFlow headers with type “OFPT_FLOW_MOD” and command “OFPFC_ADD” are there among all the packets?

Ans: 6

Wireshark抓到的狀況如下:



共有以下12個openflow headers with type “OFPT_FLOW_MOD” and command “OFPFC_ADD”



據觀察共前6個分別與後6個相同。
Value分別為

1. ARP (0x0806)
2. Unknown (0x8942) (BDDP)

3. 802.1 LLDP (0x88cc)
4. IPv4 (0x0800)
   剩下兩個則是 IN_PORT / ETH_DST / ETH_SRC (*fwd)
5. 2 / 26:3b:dc:f3:fa:1c / 76:f4:8d:0f:3f:86
6. 1 / 76:f4:8d:0f:3f:86 / 26:3b:dc:f3:fa:1c

2. What are the match fields and the corresponding actions in each “OFPT_FLOW_MOD” message?

前四種的Actions是

1. CLEAR *** Clears all actions from the datapath action set ***
2. APPLY *** Applies the action(s) immediately *** (OUTPUT) *** Output to switch port. ***

後兩種則是只有

1. APPLY (OUTPUT)

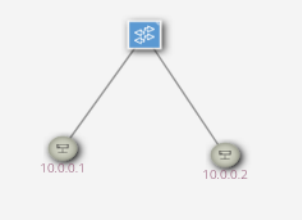
3. What are the value of timeout for each flow rule installed in s1?

前四種flow rule 的time out 是0
後兩種的flow rule(fwd)的 time out 是 10

Part 2 Install Flow Rules

於default環境下進行，如下:

```
demo@root > apps -a -s                                          16:06:08
*   6 org.onosproject.lldpprovider          2.2.0    LLDP Link Provider
*  15 org.onosproject.hostprovider          2.2.0    Host Location Provider
*  16 org.onosproject.optical-model         2.2.0    Optical Network Model
*  17 org.onosproject.openflow-base         2.2.0    OpenFlow Base Provider
*  18 org.onosproject.openflow              2.2.0    OpenFlow Provider Suite
*  55 org.onosproject.drivers               2.2.0    Default Drivers
* 140 org.onosproject.gui2                  2.2.0    ONOS GUI2
demo@root >                                                     16:06:09
```



第一次成功的json檔如下，一開始沒想太多就多給ETH_TYPE(ARP)，然後確實可以arping通，但無法ping通。

```
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "IN_PORT",
        "port": "1"
      },
      {
        "type": "ETH_TYPE",
        "ethType": "0x0806"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      }
    ]
  }
}
```

```
{
  "priority": 51000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "IN_PORT",
        "port": "2"
      },
      {
        "type": "ETH_TYPE",
        "ethType": "0x0806"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "1"
      }
    ]
  }
}
```

後再新增兩個，IPV4_DST的json (其default 要ETH_TYPE(IPv4))，即可ping通，另外嘗試取消arping的flow rule，仍可ping通，但無法arping。應該是flow rule直接指定了port，所以沒用到mac。

```
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "IN_PORT",
        "port": "2"
      },
      {
        "type": "ETH_TYPE",
        "ethType": "0x0800"
      },
      {
        "type": "IPV4_DST",
        "ip": "10.0.0.1/32"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "1"
      }
    ]
  }
}
```

```
{
  "priority": 49000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "IN_PORT",
        "port": "1"
      },
      {
        "type": "ETH_TYPE",
        "ethType": "0x0800"
      },
      {
        "type": "IPV4_DST",
        "ip": "10.0.0.2/32"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      }
    ]
  }
}
```

新增flow rule command

```
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @flows_s1-1_109550206.json 'http://localhost:8181/onos/v1/flows/of:0000
000000000001'
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @flows_s1-2_109550206.json 'http://localhost:8181/onos/v1/flows/of:0000
000000000001'
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @flows_s1-3_109550206.json 'http://localhost:8181/onos/v1/flows/of:0000
000000000001'
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @flows_s1-4_109550206.json 'http://localhost:8181/onos/v1/flows/of:0000
000000000001'
demo@SDN-NFV:~/Desktop$
```

arping、ping成功 command

```
mininet> h1 arping h2
ARPING 10.0.0.2 from 10.0.0.1 h1-eth0
Unicast reply from 10.0.0.2 [32:4F:49:2B:62:C1]  0.758ms
Unicast reply from 10.0.0.2 [32:4F:49:2B:62:C1]  0.718ms
Unicast reply from 10.0.0.2 [32:4F:49:2B:62:C1]  0.579ms
Unicast reply from 10.0.0.2 [32:4F:49:2B:62:C1]  0.533ms
Unicast reply from 10.0.0.2 [32:4F:49:2B:62:C1]  0.576ms
^CSent 5 probes (1 broadcast(s))
Received 5 response(s)
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.295 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.036 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.035 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.188 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4109ms
rtt min/avg/max/mdev = 0.032/0.117/0.295/0.107 ms
mininet>
```

後來於part4時思考，是否我的flow rule不太對，嘗試讓flow rule跟fwd的差不多:

註:下列flow rule 皆為雙向都下且mac是有給的(??只是示意)

```
flow_rule_A:
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "IN_PORT",
        "port": "1"
      },
      {
        "type": "ETH_DST",
        "mac": "??:??:??:??:??:??"
      },
      {
        "type": "ETH_SRC",
        "mac": "??:??:??:??:??:??"
      },
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      }
    ]
  }
}
```

照著fwd的flow rule去下，如flow_rule_A，發現arping不通。

```
flow_rule_B:
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "IN_PORT",
        "port": "1"
      },
      {
        "type": "ETH_TYPE",
        "ethType": "0x0806"
      },
      {
        "type": "ETH_DST",
        "mac": "??:??:??:??:??:??"
      },
      {
        "type": "ETH_SRC",
        "mac": "??:??:??:??:??:??"
      },
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      }
    ]
  }
}
```

把ETH_TYPE拿回來，如flow_rule_B，還是arping不通。

```
flow_rule_C:
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "IN_PORT",
        "port": "1"
      },
      {
        "type": "ETH_TYPE",
        "ethType": "0x0806"
      },
      {
        "type": "ETH_SRC",
        "mac": "??:??:??:??:??:??"
      },
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      }
    ]
  }
}
```

把ETH_DST拿掉，如flow_rule_C，arping通了。

```
flow_rule_D:
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x0806"
      },
      {
        "type": "ETH_SRC",
        "mac": "??:??:??:??:??:??"
      },
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      }
    ]
  }
}
```

然後覺得IN_PORT應該也不用，如flow_rule_D，arping也通了。

```
flow_rule_E:
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "ETH_SRC",
        "mac": "??:??:??:??:??:??"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      }
    ]
  }
}
```

後來覺得或許ETH_TYPE也不用，如flow_rule_E，不過失敗了，arping不通。

結論1:ETTH_TYPE一定要下。
接著仍然覺得很奇怪，為什麼不能下ETH_DST，偶然同時下了flow_rule_B、flow_rule_C，發現以下狀況，第一封broadcast是透過flow_rule_C，但接下來皆是flow_rule_B，另外還發現，如果flow_rule_C只下單向，h2 arping h1不會通。

Added	1	40	50000	0	IN_PORT:1, ETH_SRC:3A:78:D5:7A:E5:7D, ETH_TYPE:arp	imm[OUTPUT:2], cleared:false	*rest
Added	4	66	50000	0	IN_PORT:1, ETH_DST:BA:6D:DC:C4:3D:59, ETH_SRC:3A:78:D5:7A:E5:7D, ETH_TYPE:arp	imm[OUTPUT:2], cleared:false	*rest
Added	5	64	50000	0	IN_PORT:2, ETH_DST:3A:78:D5:7A:E5:7D, ETH_SRC:BA:6D:DC:C4:3D:59, ETH_TYPE:arp	imm[OUTPUT:1], cleared:false	*rest

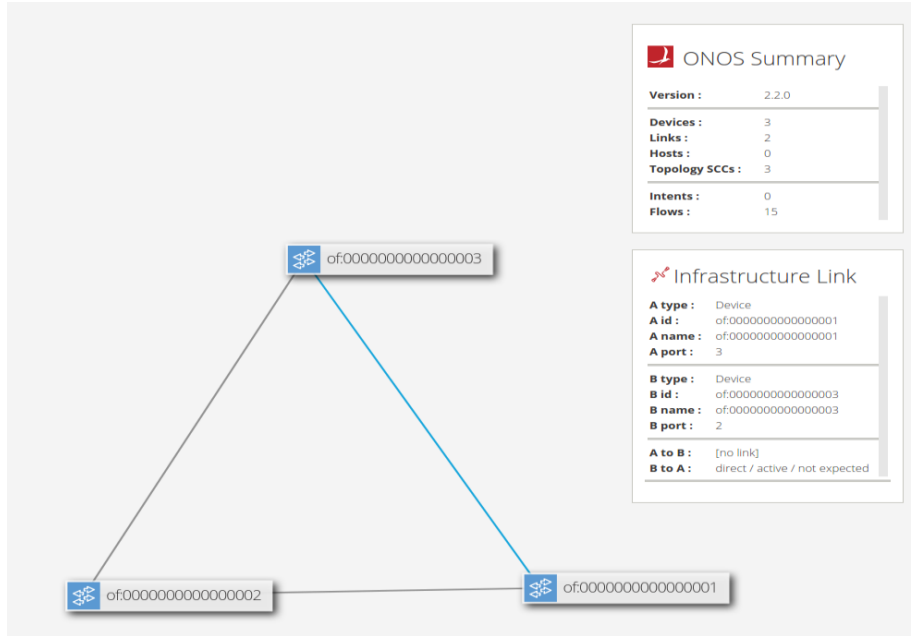
結論2:下ETH_DST的話，不會處理broadcast的封包所以無法通。
另外也嘗試了只多開ETH_TYPE:IPv4/OUTPUT:CONTROLER，能否ping通，結果無法，fwd背後有做其他事情來彌補flow rule的問題。

Part 3 Create Topology with Broadcast Storm

python script and GUI

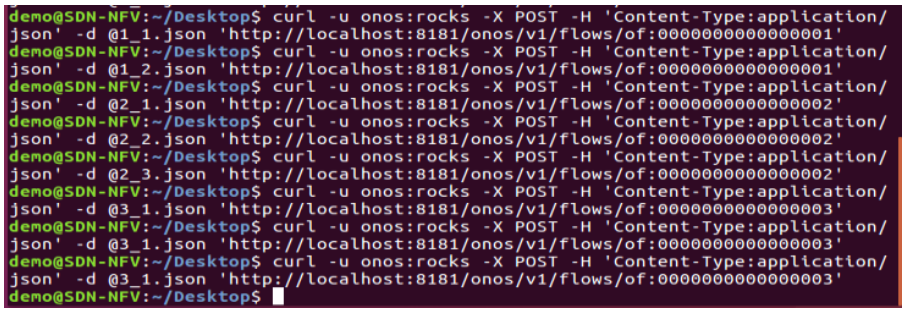
```
from mininet.topo import Topo

class Project2_Topo_109550206( Topo ):
        def __init__(self):
                Topo.__init__(self)
                #Add three switches
                s1 = self.addSwitch('s1')
                s2 = self.addSwitch('s2')
                s3 = self.addSwitch('s3')
                #Add two hosts
                h1 = self.addHost('h1')
                h2 = self.addHost('h2')
                #Add link H with S
                self.addLink(h1, s1)
                self.addLink(h2, s2)
                #Add links to make loop
                self.addLink(s1, s2)
                self.addLink(s2, s3)
                self.addLink(s3, s1)
topos = { 'topo': Project2_Topo_109550206}
```



根據GUI顯示的port去下flow rule使其傳輸路徑有loop，即會Broadcast Storm。此處flow rule就直接IN_PORT/OUTPUT$\times N$，如下：

```
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "IN_PORT",
        "port": "1"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      },
      {
        "type": "OUTPUT",
        "port": "3"
      }
    ]
  }
}
```



```
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @1_1.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @1_2.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @2_1.json 'http://localhost:8181/onos/v1/flows/of:0000000000000002'
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @2_2.json 'http://localhost:8181/onos/v1/flows/of:0000000000000002'
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @2_3.json 'http://localhost:8181/onos/v1/flows/of:0000000000000002'
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @3_1.json 'http://localhost:8181/onos/v1/flows/of:0000000000000003'
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @3_1.json 'http://localhost:8181/onos/v1/flows/of:0000000000000003'
demo@SDN-NFV:~/Desktop$ curl -u onos:rocks -X POST -H 'Content-Type:application/
json' -d @3_1.json 'http://localhost:8181/onos/v1/flows/of:0000000000000003'
demo@SDN-NFV:~/Desktop$
```

我設計的路線為:

h1 → s1

h2 → s2

s1 → h1

s1 → s2

s1 → s3

s2 → h2

s2 → s1

s3 → h2

When h1 ping h2

封包路線:

h1 → s1 → s2 → h2

s1→ s3 → s2 → h2

s1→ s3 → s2 → s1 → s3 (loop)

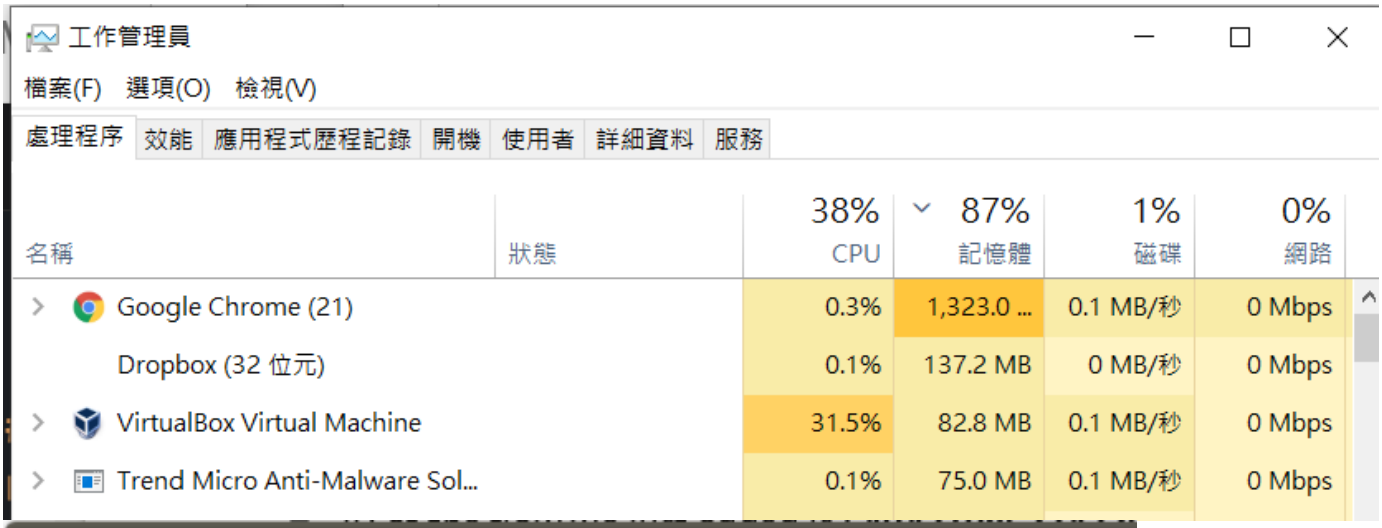
When h2 ping h1

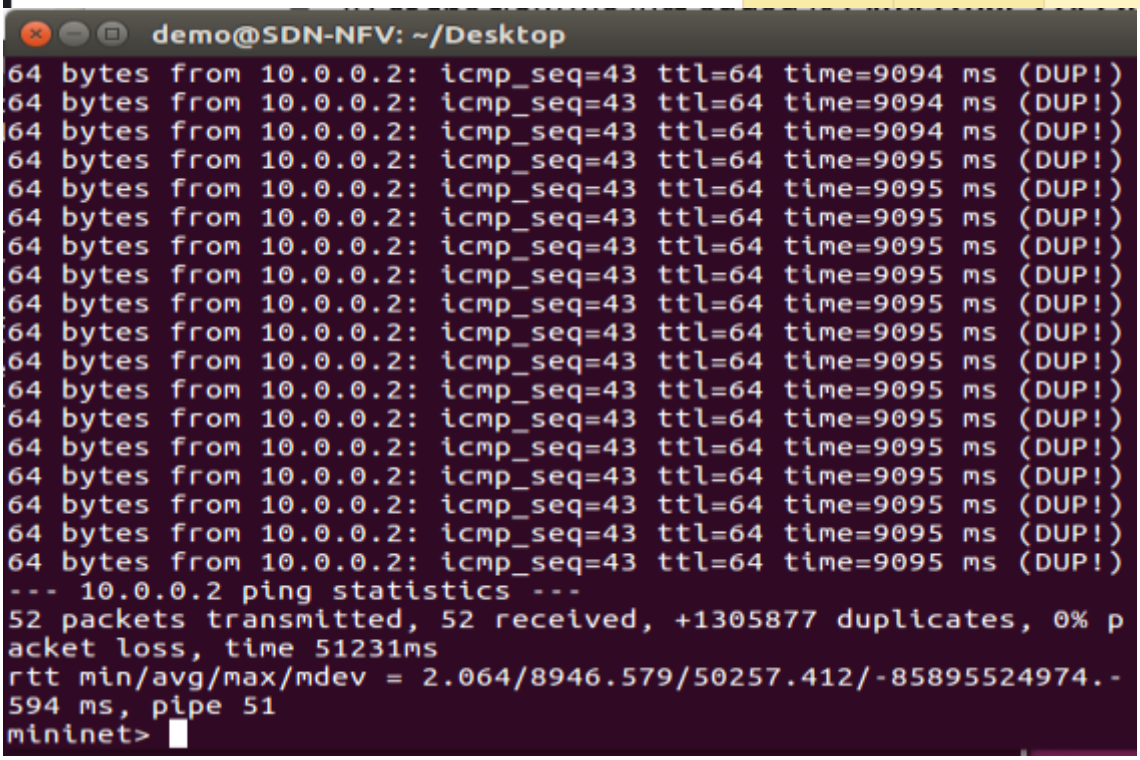
封包路線:

h2 → s2 → s1 → h1

s1→ s3 → s2 → s1 (loop)

When broadcast:

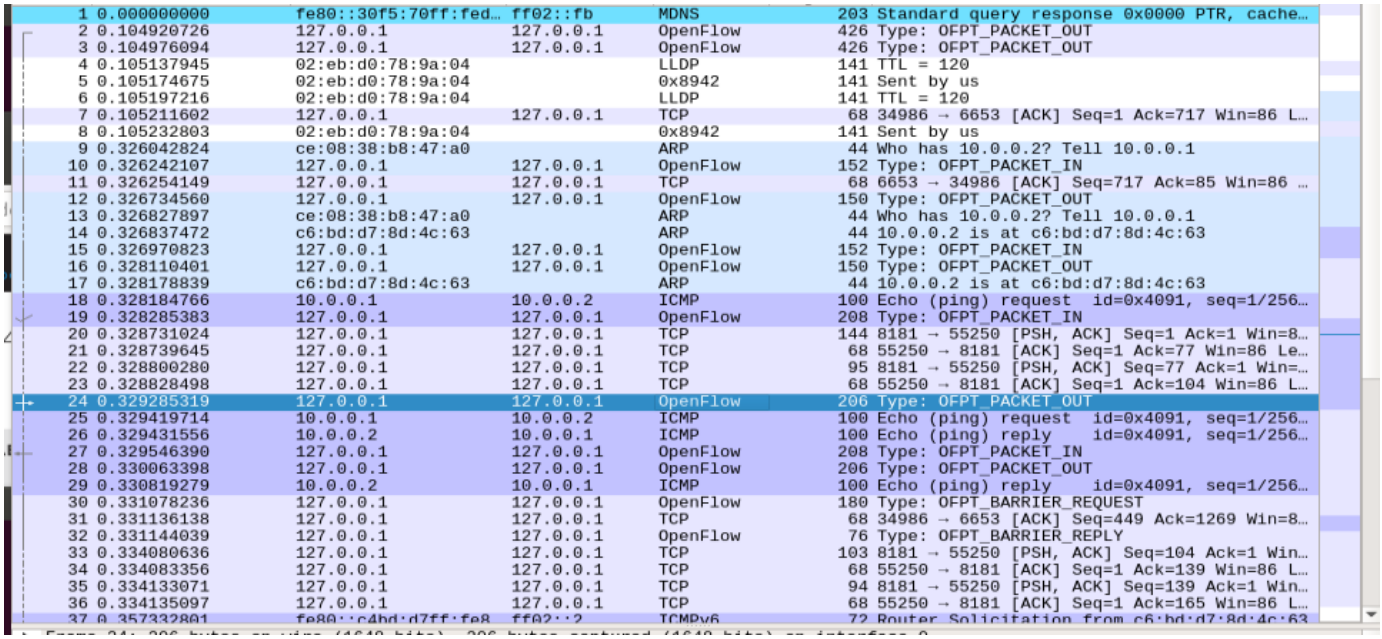




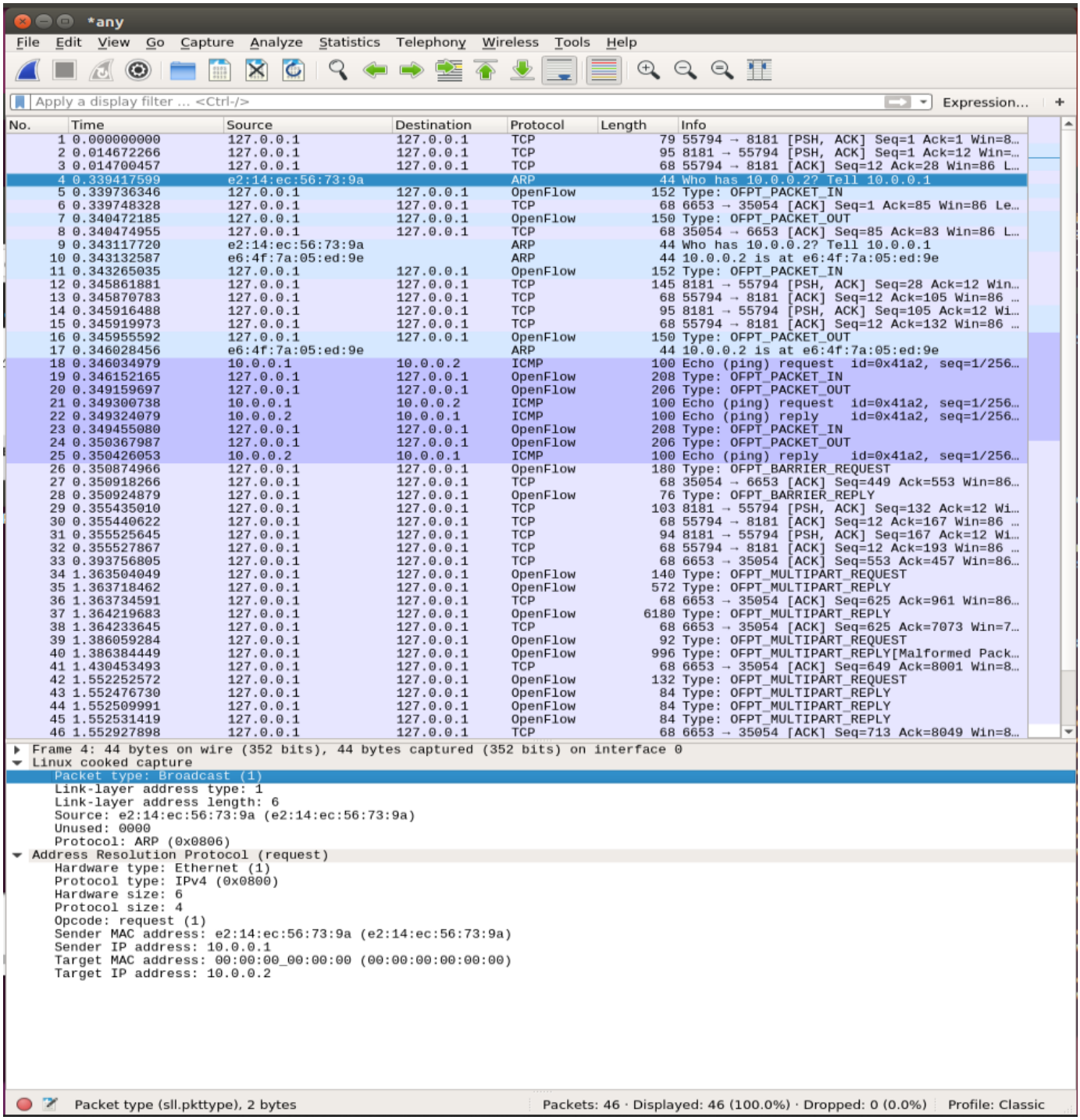
註:(DUP!)產生速度快過顯示

Part 4 Trace ReactiveForwarding

下兩圖皆為已開啟fwd後，再開啟wireshark後，h1 ping h2 -c 1之後，停止wireshark的封包



1	0.000000000	fe80::30f5:70ff:fed…	ff02::fb	MDNS	203 Standard query response 0x0000 PTR, cache…
2	0.104920726	127.0.0.1	127.0.0.1	OpenFlow	426 Type: OFPT_PACKET_OUT
3	0.104976094	127.0.0.1	127.0.0.1	OpenFlow	426 Type: OFPT_PACKET_OUT
4	0.105137945	02:eb:d0:78:9a:04		LLDP	141 TTL = 120
5	0.105174675	02:eb:d0:78:9a:04		0x8942	141 Sent by us
6	0.105197216	02:eb:d0:78:9a:04		LLDP	141 TTL = 120
7	0.105211602	127.0.0.1	127.0.0.1	TCP	68 34986 → 6653 [ACK] Seq=1 Ack=717 Win=86 L…
8	0.105232803	02:eb:d0:78:9a:04		0x8942	141 Sent by us
9	0.326042824	ce:08:38:b8:47:a0		ARP	44 Who has 10.0.0.2? Tell 10.0.0.1
10	0.326242107	127.0.0.1	127.0.0.1	OpenFlow	152 Type: OFPT_PACKET_IN
11	0.326254149	127.0.0.1	127.0.0.1	TCP	68 6653 → 34986 [ACK] Seq=717 Ack=85 Win=86 …
12	0.326734560	127.0.0.1	127.0.0.1	OpenFlow	150 Type: OFPT_PACKET_OUT
13	0.326827897	ce:08:38:b8:47:a0		ARP	44 Who has 10.0.0.2? Tell 10.0.0.1
14	0.326837472	c6:bd:d7:8d:4c:63		ARP	44 10.0.0.2 is at c6:bd:d7:8d:4c:63
15	0.326970823	127.0.0.1	127.0.0.1	OpenFlow	152 Type: OFPT_PACKET_IN
16	0.328110401	127.0.0.1	127.0.0.1	OpenFlow	150 Type: OFPT_PACKET_OUT
17	0.328178839	c6:bd:d7:8d:4c:63		ARP	44 10.0.0.2 is at c6:bd:d7:8d:4c:63
18	0.328184766	10.0.0.1	10.0.0.2	ICMP	100 Echo (ping) request id=0x4091, seq=1/256…
19	0.328285383	127.0.0.1	127.0.0.1	OpenFlow	208 Type: OFPT_PACKET_IN
20	0.328731024	127.0.0.1	127.0.0.1	TCP	144 8181 → 55250 [PSH, ACK] Seq=1 Ack=1 Win=8…
21	0.328739645	127.0.0.1	127.0.0.1	TCP	68 55250 → 8181 [ACK] Seq=1 Ack=77 Win=86 Le…
22	0.328800280	127.0.0.1	127.0.0.1	TCP	95 8181 → 55250 [PSH, ACK] Seq=77 Ack=1 Win=…
23	0.328828498	127.0.0.1	127.0.0.1	TCP	68 55250 → 8181 [ACK] Seq=1 Ack=104 Win=86 L…
24	0.329285319	127.0.0.1	127.0.0.1	OpenFlow	206 Type: OFPT_PACKET_OUT
25	0.329419714	10.0.0.1	10.0.0.2	ICMP	100 Echo (ping) request id=0x4091, seq=1/256…
26	0.329431556	10.0.0.2	10.0.0.1	ICMP	100 Echo (ping) reply id=0x4091, seq=1/256…
27	0.329546390	127.0.0.1	127.0.0.1	OpenFlow	208 Type: OFPT_PACKET_IN
28	0.330063398	127.0.0.1	127.0.0.1	OpenFlow	206 Type: OFPT_PACKET_OUT
29	0.330819279	10.0.0.2	10.0.0.1	ICMP	100 Echo (ping) reply id=0x4091, seq=1/256…
30	0.331078236	127.0.0.1	127.0.0.1	OpenFlow	180 Type: OFPT_BARRIER_REQUEST
31	0.331136138	127.0.0.1	127.0.0.1	TCP	68 34986 → 6653 [ACK] Seq=449 Ack=1269 Win=8…
32	0.331144039	127.0.0.1	127.0.0.1	OpenFlow	76 Type: OFPT_BARRIER_REPLY
33	0.334080636	127.0.0.1	127.0.0.1	TCP	103 8181 → 55250 [PSH, ACK] Seq=104 Ack=1 Win…
34	0.334083356	127.0.0.1	127.0.0.1	TCP	68 55250 → 8181 [ACK] Seq=1 Ack=139 Win=86 L…
35	0.334133071	127.0.0.1	127.0.0.1	TCP	94 8181 → 55250 [PSH, ACK] Seq=139 Ack=1 Win…
36	0.334135097	127.0.0.1	127.0.0.1	TCP	68 55250 → 8181 [ACK] Seq=1 Ack=165 Win=86 L…
37	0.357332801	fe80::c4bd:d7ff:fe8…	ff02::2	ICMPv6	72 Router Solicitation from c6:bd:d7:8d:4c:63



1. h1丟出一個ARP (Who has 10.0.0.2? Tell 10.0.0.1)
2. s1收到後miss (Flow table沒有對應的方法，s1 does not know who is 10.0.0.2)
3. s1丟一個Open Flow去問c1 (OFPT_PACKET_IN: IN_PORT:1)
4. c1回s1 (OFPT_PACKET_OUT: OUTPUT:2)
5. s1把該ARP從Port 2轉給h2
6. h2收到ARP，因h2即為10.0.0.2，因此回一個ARP (10.0.0.2 at ??:??:??:??:??:??)
7. s1收到後，仍然miss (還沒有對應的flow rule下來)
8. s1丟一個Open Flow去問c1 (OFPT_PACKET_IN: IN_PORT:2)
9. c1回s1 (OFPT_PACKET_OUT: OUTPUT:1)
10. s1把該ARP從Port 1轉給h1
11. h1收到，得知10.0.0.2在哪個mac，並丟ICMP過去

12. s1收到後，仍然miss (還沒有對應的flow rule下來)
13. s1丟一個Open Flow去問c1 (OFPT_PACKET_IN: IN_PORT:1)
14. c1回s1 (OFPT_PACKET_OUT: OUTPUT:2)
15. s1把該ICMP從Port 2轉給h2
16. h2收到first ICMP request

What you've learned or solved

這次Lab學的東西有點多，實際去抓封包、研究、思考fwd在幹嘛，具體而言還是有一些模稜兩可不確定的地方，但整體而言多了解很多東西，也與老師上課講的架構、概念、方法有對到。一開始做part2還滿卡的，不知道怎麼下手，先隨便猜，然後過了，接著做part3時，覺得滿有趣的(可能是看到跑超快DUP和塊當掉的VM覺得有趣?!)，然後開始思考part4時，回去想part2的東西，這塊覺得是最有成就感的地方，相對於一開始更完整了解flow rule各個match filed/intruction/action的功能、意義。